By Falko Timme
Published: 2008-08-31 19:31

Triggering Commands On File/Directory Changes With Incron

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 08/21/2008

This guide shows how you can install and use **incron** on a Debian Etch system. Incron is similar to cron, but instead of running commands based on time, it can trigger commands when file or directory events occur (e.g. a file modification, changes of permissions, etc.).

This document comes without warranty of any kind! I do not issue any guarantee that this will work for you!

1 Installing Incron

Incron is available in the *etch-backports* repository, so we have to add the following line to */etc/apt/sources.list*:

```
vi /etc/apt/sources.list
```

```
[...]
deb http://www.backports.org/debian etch-backports main contrib non-free
[...]
```

Let's import the backports.org archive key into apt...

```
wget -O - http://backports.org/debian/archive.key | apt-key add -
```

... and run

```
apt-get update
```

The *etch-backports* repository is inactive by default, so to install incron from it, we must use the following command:

```
apt-get -t etch-backports install incron
```

2 Using Incron

Incron usage is very much like cron usage. You have the *incrontab* command that let's you list (*-l*), edit (*-e*), and remove (*-r*) incrontab entries. To learn more about it, see

```
man incrontab
```

There you also find the following section:

If /etc/incron.allow exists only users listed here may use incron. Otherwise if /etc/incron.deny exists only users NOT listed here may use incron. If none of these files exists everyone is allowed to use incron. (Important note: This behavior is insecure and will be probably changed to be compatible with the style used by ISC Cron.) Location of these files can be changed in the configuration.

This means if we want to use incrontab as root, we must either delete */etc/incron.allow* (which is unsafe because then every system user can use incrontab)...

```
rm -f /etc/incron.allow
```

... or add root to that file (recommended):

```
vi /etc/incron.allow
```

```
root
```

Before you do this, you will get error messages like this one when trying to use incrontab:

```
server1:~# incrontab -l
  user 'root' is not allowed to use incron
server1:~#
```

Afterwards it works:

```
server1:~# incrontab -l
  no table for root
server1:~#
```

We can use

```
incrontab -e
```

to create incron jobs. Before we do this, we take a look at

```
man 5 incrontab
```

because it explains the format of the crontabs. Basically the format is as follows...

```
<path> <mask> <command>
```

...where <path> can be a directory (meaning the directory and/or the files directly in that directory (not files in subdirectories of that directory!) are

watched) or a file.

<*mask*> can be one of the following:

```
IN_ACCESS           File was accessed (read) (*)
IN_ATTRIB           Metadata changed (permissions, timestamps, extended attributes, etc.) (*)
IN_CLOSE_WRITE      File opened for writing was closed (*)
IN_CLOSE_NOWRITE    File not opened for writing was closed (*)
IN_CREATE           File/directory created in watched directory (*)
IN_DELETE           File/directory deleted from watched directory (*)
IN_DELETE_SELF           Watched file/directory was itself deleted
IN_MODIFY           File was modified (*)
IN_MOVE_SELF        Watched file/directory was itself moved
IN_MOVED_FROM       File moved out of watched directory (*)
IN_MOVED_TO         File moved into watched directory (*)
IN_OPEN             File was opened (*)
```

When monitoring a directory, the events marked with an asterisk (*) above can occur for files in the directory, in which case the name field in the returned event data identifies the name of the file within the directory.

The *IN_ALL_EVENTS* symbol is defined as a bit mask of all of the above events. Two additional convenience symbols are *IN_MOVE*, which is a combination of *IN_MOVED_FROM* and *IN_MOVED_TO*, and *IN_CLOSE* which combines *IN_CLOSE_WRITE* and *IN_CLOSE_NOWRITE*.

The following further symbols can be specified in the mask:

```
 IN_DONT_FOLLOW      Don't dereference pathname if it is a symbolic link
IN_ONESHOT          Monitor pathname for only one event
IN_ONLYDIR          Only watch pathname if it is a directory
```

Additionaly, there is a symbol which doesn't appear in the inotify symbol set. It is *IN_NO_LOOP*. This symbol disables monitoring events until the current one is completely handled (until its child process exits).

<*command*> is the command that should be run when the event occurs. The following wildards may be used inside the command specification:


```
$$   dollar sign
$@   watched filesystem path (see above)
$#   event-related file name
$%   event flags (textually)
$&   event flags (numerically)
```

If you watch a directory, then *$@* holds the directory path and *$#* the file that triggered the event. If you watch a file, then *$@* holds the complete path to the file and *$#* is empty.

If you need the wildcards but are not sure what they translate to, you can create an incron job like this:

```
/tmp/ IN_MODIFY echo "$$ $@ $# $% $&"
```

Then you create or modify a file in the */tmp* directory and take a look at */var/log/syslog* - this log shows when an incron job was triggered, if it succeeded or if there were errors, and what the actual command was that it executed (i.e., the wildcards are replaced with their real values).

```
tail /var/log/syslog
```

```
...
Aug 21 17:26:50 server1 incrond[7111]: (root) CMD (echo "$ /tmp huhu IN_CREATE 256")
```

In this example I've created the file */tmp/huhu*; as you see *$@* translates to */tmp*, *$#* to *huhu*, *$%* to *IN_CREATE*, and *$&* to *256*.

Now enough theory. Let's create our first incron jobs. I'd like to monitor the file */etc/apache2/apache2.conf* and the directory */etc/apache2/vhosts/*, and whenever there are changes, I want incron to restart Apache. This is how we do it:

```
incrontab -e
```

```
/etc/apache2/apache2.conf IN_MODIFY /etc/init.d/apache2 restart
/etc/apache2/vhosts/ IN_MODIFY /etc/init.d/apache2 restart
```

That's it. For test purposes you can modify your Apache configuration and take a look at */var/log/syslog*, and you should see that incron restarts Apache.

To list all defined incron jobs, you can run

```
incrontab -l
```

```
server1:~# incrontab -l
  /etc/apache2/apache2.conf IN_MODIFY /etc/init.d/apache2 restart
  /etc/apache2/vhosts/ IN_MODIFY /etc/init.d/apache2 restart
server1:~#
```

To delete all incron jobs of the current user, run

```
incrontab -r
```

```
server1:~# incrontab -r
  removing table for user 'root'
  table for user 'root' successfully removed
server1:~#
```

3 Links

- incron: ***http://inotify.aiken.cz/?section=incron&page=about&lang=en***